REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
20-MAY-2002	Conference Proceedings, (not refereed)	

4. TITLE AND SUBTITLE
The 2-3TR-tree, A Trajectory-Oriented Index Structure For Fully Evolving Valid-Time Spatio-Temporal Datasets

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0602435N

6. AUTHOR(S)
J. Givaudan M. ABDELGUERFI KEVIN B SHAW ROY VICTOR LADNER

5d. PROJECT NUMBER
74-6731-02

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
Marine Geoscience Division
Stennis Space Center, MS 39529-5004

8. REPORTING ORGANIZATION REPORT NUMBER
NRL/PP/7440--02-1015

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research and Naval Research Lab
800 North Quincy Street Code 7440.2
Arlington, VA 22217 Stennis Space Center, MS

10. SPONSOR/MONITOR'S ACRONYM(S)
ONR & NRL

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release,distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Supporting large volumes of multi-dimensional data is an inherent characteristic of modern database applications, such as Geographical Information Systems (GIS), Computer Aided design (CAD), and Image and Multimedia Databases. Such databases need underlying systems with extended features like query languages, data models, and indexing methods, as compared to traditional databases, mainly because of the complexity of representing and retrieving data. The presented work deals with access methods for databases that accurately model the real world. More precisely, the focus is on index structures that can capture the time varying nature of moving objects, namely spatio-temporal structures. A new taxonomy to classify these structures has been defined according to dataset characteristics and query requirements. Then, a new spatio-temporal access method, the 2-3TR-tree, has been designed to process specific datasets and fulfill specific query requirements that no other existing spatio-temporal index could handle.

15. SUBJECT TERMS
Spatio-temporal, indexing, R-tree, Taxonomy

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a NAME OF RESPONSIBLE PERSON / 19b. TELEPHONE NUMBER *(Include area code)*
unclassified	unclassified	unclassified	Unlimited	20	Kevin Shaw / 228-688-4197

Standard Form 298 (Rev. 8/98)

20021025 325

The 2-3TR-tree, A Trajectory-Oriented Index Structure for Fully Evolving Valid-Time Spatio-Temporal Datasets

Givaudan, J., Abdelguerfi, M.
{jgivauda,mahdi}@cs.uno.edu
Computer Science Department, University of New Orleans
Tel: (504) 280-7107
Shaw, K., Ladner, R.
Naval Research Laboratory, Stennis Space Center, MS

Abstract- Supporting large volumes of multi-dimensional data is an inherent characteristic of modern database applications, such as Geographical Information Systems (GIS), Computer Aided design (CAD), and Image and Multimedia Databases. Such databases need underlying systems with extended features like query languages, data models, and indexing methods, as compared to traditional databases, mainly because of the complexity of representing and retrieving data. The presented work deals with access methods for databases that accurately model the real world. More precisely, the focus is on index structures that can capture the time varying nature of moving objects, namely spatio-temporal structures. A new taxonomy to classify these structures has been defined according to dataset characteristics and query requirements. Then, a new spatio-temporal access method, the 2-3TR-tree, has been designed to process specific datasets and fulfill specific query requirements that no other existing spatio-temporal index could handle.

Key Words – Spatio-temporal, indexing, R-tree, Taxonomy.

1. Spatio-Temporal R-tree-based Structures

The primary goal of a spatio-temporal databases [LASA02] is the accurate modeling of the real world; that is a dynamic world, which involves objects whose position, shape and size change over time. Real life examples that need to handle spatio-temporal data include storage and manipulation of ship and plane trajectories, fire or hurricane front monitor and weather forecast. Geographical information systems are also a source for spatio-temporal data. Thus, spatio-temporal databases are used in many applications; therefore, in this section we are interested in structures that can index efficiently such databases. We deal with data points, since in many applications the size and shape of an object do not matter, only its position does.

To construct a spatio-temporal index structure the approach in current research is to extend existing spatial access methods in order to keep their spatial properties. There are two major trends considering the indexing of spatial data. The first trend is to use space-driven structures: these are indices based on the partitioning of the embedding two-dimensional space into rectangular cells, independently of the distribution of the objects in the 2D plane. The most popular space-driven structures are the Quadtree and its variants [FIBE74].

The second trend is to use data-driven structures to index spatial data. Data-driven structures are organized by partitioning the set of objects, and not the embedding space. The most used data driven structures are the R-tree [GUTT84] and its variants [SELL87, BECK90, and KAFA94]. R-trees are generally more efficient for non-uniform data than Quadtrees. Indeed, a concentration of data points in the 2D space results in a very long branch in a Quadtree, which in turn increases drastically the processing time for insertion, deletion and spatial searching in this area. This does not happen in R-trees since they are height-balanced. Therefore, we will focus on R-tree-based structures.

In this section, we first define a taxonomy of spatio-temporal access methods that brings out the different dimensions to be considered in designing a spatio-temporal index structure. Then, we review the existing R-tree-based structures that handle spatio-temporal data and classify them according to the newly defined taxonomy.

2.Taxonomy

The taxonomy proposed in this paper is made up of the different dimensions that have to be considered in the design and evaluation of a spatio-temporal access method. These are *eight* distinct dimensions that are related to the dataset characteristics, the query requirements, discrimination, insertion/split strategy and version duplication. It should be noted that, to our knowledge, there exists only one other taxonomy for spatio-temporal structures that has been reported in [TS98]. The classification carried out in [TS98] does not include as many spatio-temporal structures as those presented in this paper, and does not cover all the dimensions specified by our new taxonomy. As a consequence, some existing spatio-temporal indices cannot be differentiated using the taxonomy in [TS98].

The eight dimension distinct dimensions of our taxonomy are presented below:

Dataset Characteristics

1. Time Dimension

Different temporal aspects can be considered when storing temporal data:

Valid-time: This is the time when the fact is true in the modeled reality. The valid-time of a fact can in the past, or in the future and can be changed freely.

Transaction-time: This is the time when the fact is current in the database and may be retrieved. The transaction-time of a fact cannot extend beyond the current time nor can it be changed. Insertions and deletions can only happen at the current time, in other words there is no update in the past. Moreover, deletions are logical: the actual records are not physically removed from the database.

Bitemporal data: Data with both valid and transaction-time.

2. *Time Evolution*

Datasets can be constituted of positions or parts of trajectories (that can be interpolated). Therefore, two aspects of time evolution can be considered:

Discrete: Only positions of objects at different timestamps are stored. No information can be retrieved between two timestamps, or only the position at the previous timestamp.

Continuous: Continuous in this context does not mean that we can retrieve the exact position of an object at any timestamp. Instead, it means that trajectories or trajectory segments are stored. They may be interpolated, i.e. the positions of an object are recorded, and the trajectory between them is interpolated (in most cases linearly). Therefore, the database can be queried for any value of time.

3. *Data Evolution*

This dimension deals with the mobility in time and in space (location) of the data objects. The dataset can be:

Moving: The cardinality of the database, i.e. the number of object instances, does not change over time, but the database objects can change their location.

Growing: The cardinality of the database can change, but the objects stay at the same location.

Fully Evolving: The cardinality of the database can change and the objects can change location.

4. *Data Acquisition*

There are different ways for the object instances to be inserted in the database or updated. The acquisition of the data can be:

Static: The data is known before hand and no insertions or updates are allowed.

Chronological: Only current instances are dynamically inserted or updated. In transaction-time databases, if data acquisition is not static, it is chronological and past cannot be changed.

Dynamic: Insertions and updates to any object referred to any timestamp are allowed.

Query Requirements

5. *Query Types*

In addition to classical spatial range queries, different types of queries may be required by an application. Based on this, an application can be:

Historical information oriented: In this type of applications, access methods aim at the retrieval of historical spatio-temporal information about objects that evolve. This involves queries of the form "Find all objects that have lied within a specific area (or at a

specific point), during a specific time interval (or at a specific timestamp)". Time slice or timestamp queries retrieve all objects that intersect a window at a specific timestamp. Time interval queries include several (usually consecutive) timestamps.

Trajectory oriented: Certain applications need to store motion patterns of objects. The goal in this case is to retrieve the line segments or position points representing the trajectories of certain objects. We will especially focus on the retrieval of trajectories carried out by selecting a segment of the trajectory using a spatio-temporal range, and then identifying the remaining segments or points that are part of the trajectory.

Combined: This type of applications combines both historical information retrieval and trajectory retrieval.

Spatial/Temporal Discrimination

6.Discrimination

The discrimination capability of an access method deals with the workspace dimensions/properties used to index and retrieve the data in the index structure. The more discrimination there is, the better the range query performance is. In the context of spatio-temporal access methods, there are three possible levels of discrimination:

Spatial: Data is indexed and retrieved according to spatial dimensions.

Temporal: Data is indexed and retrieved according to the time dimension.

Spatio-temporal: Data is indexed and retrieved according to both spatial and time dimensions.

Insertion/Split

7.Insertion/Split Strategy

This dimension deals with the construction of the index structure. It represents the strategy that is applied when choosing the insertion path or splitting a node that overflows. The strategy adopted by an index structure plays a role in query processing and affects the performance of certain query types. There are three different strategies that can be emphasized:

Least Enlargement: To insert a new entry in the index, the path is chosen with respect to the least enlargement criterion. In other words, we follow the nodes whose MBBs need the least enlargement to insert the new entry. Similarly, the entries of an overflowing node are split into two nodes according to the least enlargement criterion. This is the strategy used by the R-tree.

Table 1. Classification of Existing Spatio-Temporal Structures

Structure / Dimension	RT-tree	3D R-tree (Version 1)	3D R-tree (Version 2)	3D R-tree (Version 3)	2+3 R-tree	HR-tree	MV3R-tree	STR-tree	TB-tree
Time Dimension	Transaction-time	Valid-time	Valid-time	Valid-time	Valid-tim	Transaction-time	Transaction-time	Transaction-time	Transaction-time
Time Evolution	Discrete	Continuous	Continuous	Discrete	Discrete	Discrete	Discrete	Continuous	Continuous
Data Evolution	Fully Evolving	Growing	Fully Evolving	Fully Evolving	Fully Evolving	Fully Evolving	Fully Evolving	Fully Evolving	Fully Evolving
Data Acquisition	Chronological	Static	Static	Static	Dynamic	Chronological	Chronological	Chronological	Chronological
Query Type	Historical Information Oriented	Combined	Historical Information Oriented	Historical Information Oriented	Historical Information Oriented	Historical Information Oriented	Historical Information Oriented	Combined	Trajectory Oriented
Discrimination	Spatial	Spatio-temporal	Spatio-temporal	Spatio-temporal	Spatio-temporal	Spatio-temporal	Spatio-temporal	Spatio-temporal	Temporal
Insertion/Split Strategy	Minimum Overlap or Linear Ordering	Minimum Overlap or Linear Ordering	Minimum Overlap or Linear Ordering	Minimum Overlap or Linear Ordering	Minimum Overlap or Linear Ordering	Least Enlargement	Minimum Overlap or Linear Ordering	Trajectory Preservation	Trajectory Preservation
Version Duplication	No Duplication	No Duplication	No Duplication	No Duplication	No Duplication	Duplication	Duplication	No Duplication	No Duplication

Minimum Overlap: The insertion and split algorithms assign an entry to the nodes that result in a minimum overlap between the MBBs of the nodes at the same level. Since there is less overlap than with the least enlargement strategy, fewer paths may need to be searched for historical information retrievals, and thus, search performance is increased. This is the strategy used by R*-trees [BECK90].

Linear Ordering: The insertion and split algorithms assign an entry according to a linear ordering defined by a space filling curve. This strategy improves node utilization, and therefore historical information retrievals. So far, the Minimum Overlap and the Linear Ordering are the most efficient insertion/split strategies.

Trajectory Preservation: The insertion and split algorithms group the entries according to the trajectories they belong to. This strategy improves trajectory retrievals to the detriment of historical information retrievals since it usually involves more overlap.

Version Duplication

8. Version Duplication

This dimension indicates whether the index structure contains multiple copies of the same object at the same positions but at different timestamps. This phenomenon usually occurs with overlapping and multi-version techniques. Version duplication is to be considered if storage requirement is an important parameter for the application that uses the access method. Indeed, data duplication will increase the size of the index structure. Obviously, the two possible values for version duplication are:

Duplication: Version duplication is involved.

No Duplication: No version duplication is involved

We now classify existing spatio-temporal R-tree based data structures according to the previously defined taxonomy. As stated previously, we focus on R-tree-based structures. However, it should be noted that currently there is one spatio-temporal index based on the Quadtree: the Overlapping Linear Quadtree [TZO98]. Regarding the R-tree-based structures, there is a limited number of proposals: the RT-tree [XULU90], the 3D R-tree[1] [VAZ96], the 2+3 R-tree [NST99], the HR-tree [NAS98], the MV3R-tree [YUPA00], and the STR-tree and the TB-tree [PJT00]. Table 1classifies these existing R-tree structures using our newly defined taxonomy.

Table 2 shows that existing spatio-temporal indices cannot handle application categories that have the following characteristics: valid-time database, dynamic data acquisition, fully evolving, retrieval of both historical information and trajectories (*combined* queries), *spatio-temporal* discrimination and a minimum overlap or linear ordering insertion/split strategy.

[1] It is noted that 3 versions of the 3D-Tree are presented in [VAZ96].

Table 2. Evaluation of Existing Spatio-Temporal Structures

Requirement / Structure	Valid-Time	Fully Evolving Data	Dynamic Data Acquisition	Combined Queries	Spatio-Temporal Discrimination	Minimum Overlap or Linear Ordering Strategy
RT-tree	No	Yes	No	No	No	Yes
3D R-tree (version 1)	Yes	No	No	Yes	Yes	Yes
3D R-tree (version 2)	Yes	Yes	No	No	Yes	Yes
3D R-tree (version 3)	Yes	Yes	No	No	Yes	Yes
2+3R-tree	Yes	Yes	Yes	No	Yes	Yes
HR-tree	No	Yes	No	No	Yes	No
MV3R-tree	No	Yes	No	No	Yes	Yes
STR-tree	No	Yes	No	Yes	Yes	No
TB-tree	No	Yes	No	No	No	No

It is emphasized that many application categories of critical importance have the above requirements, and thus cannot be handled by existing spatio-temporal structures.

3. The 2-3 Trajectory R-tree

This paragraph proposes a spatio-temporal access method, the 2-3 Trajectory R-tree (2-3TR-tree), which meets the requirements defined above. The 2-3TR-tree is based on the 2+3R-tree and the TB-tree. Like the 2+3 R-tree, the 2-3TR-tree consists of two R-tree indices. Any derivative of the R-tree can be used. More particularly, two R*-trees or two Hilbert R-trees [KAFA94] can be used,

which allows applying a *minimum overlap* or *linear ordering* insertion/split strategy. In the following we refer to these indices as R-trees.

In the 2-3TR-tree, one R-tree indexes two-dimensional points, and the other one indexes three-dimensional points or lines. The two-dimensional points represent the current spatial information about the data, i.e. the current objects' positions. The current positions are kept along with their start time. The three-dimensional points or lines represent the historical information about the data. A three-dimensional point corresponds to a position that was held by an object during only one timestamp. A three-dimensional line represents a position that was held during a time interval. Thus, the indexed data are not trajectory segments but the objects' positions. As a result, there is no dead space involved by moving objects like in the 3D R-tree versions 1 and 2. Therefore, the 2-3TR-tree handles *fully evolving* data.

While the end time of an object's position is unknown, it is indexed under the two-dimensional R-tree. Once the end time of an object's current position is known, the position is inserted in the three-dimensional R-tree along with its start and end times, and removed from the two-dimensional R-tree. Therefore, no open cube is created when an object whose movement is unknown is inserted. Moreover, data can be inserted or updated in past states. As a consequence, the 2-3TR-tree allows *dynamic* data acquisition and handles *valid-time* databases. Furthermore, the three-dimensional R-tree offers *spatio-temporal* discrimination.

Finally, an additional difference with the 2+3R-tree is that when a point (x_0, y_0) indexed under the two dimensional R-tree changes position at t_j, the segment $[(x_0, y_0, t_i) (x_0, y_0, t_{j-1})]$ is inserted in the three-dimensional R-tree, instead of the segment $[(x_0, y_0, t_i) (x_0, y_0, t_j)]$. This slightly reduces overlap (at t_j the object is indexed only once) and does not involve any loss of information since we consider discretely moving points.

The data structure of the 2-3TR-tree is based on that of the TB-tree. Each position held by a data object is represented by an object, *FeatureSnapshot*, which contains the position's spatial coordinates as well as its time information. The time information can be a single timestamp or the start and end times of a time interval. A *FeartureSnapshot object* also contains an array associating certain timestamps to corresponding sets of attributes. If a data object remains at the same location during a time interval, different sets of data information can then be recorded in the array. If a data object holds a position during only one timestamp, the array has only one entry.

Each *FeatureSnapshot* has a pointer to a *FeatureTrajectory object*. A *FeatureTrajectory object* is a superstructure linking all the locations where a specific object has been. It contains an array that is sorted with respect to time and associates timestamps to pointers to the

corresponding *FeatureSnapshot* objects. If a *FeatureSnapshot* object has a duration (i.e. the position it represents is held during a time interval), the associated timestamp in its *FeatureTrajectory* is its start time.

In the 2-3TR-tree, range queries are performed as dictated by the R-tree structure chosen for the two indices. The *FeatureTrajectory object* permits the efficient retrieval of a data object's trajectory from one of its positions. Indeed, once a specific *FeatureSnapshot F* is obtained, the array in the *FeatureTrajectory* pointed by *F* is used to get the successive positions of the data object. Therefore, the 2-3TR-tree handles *combined* queries. In addition, the *FeatureTrajectory*'s array allows retrieving partial trajectories, i.e. a data object's positions during a specific time interval.

The advantages of the 2-3TR-tree are summarized below:

- The 2-3TR-tree handles *valid-time* and *fully evolving* datasets.
- It supports *dynamic* data acquisition.
- It uses a *minimum overlap* or a *linear ordering* insertion/split strategy and offers *spatio-temporal* discrimination: therefore, it efficiently retrieves historical information.
- It does not involve *version duplication.*

4. Performance Evaluation

This section strives to give a performance evaluation for the 2-3TR-tree. We are interested in evaluating the cost of historical information retrievals (spatio-temporal range queries), since trajectories will be straightforwardly retrieved from historical information retrievals by using the modified data structure of the 2-3TR-tree.

The cost model for the 2-3TR-tree will be derived using the following notations:

Symbol	Definition
R_{3d}	Three-dimensional R-tree
R_{2d}	Two-dimensional R-tree
f_{3d}	Average three-dimensional R-tree node fanout
f_{2d}	Average two-dimensional R-tree node fanout
h_{3d}	Height of the three-dimensional R-tree
h_{2d}	Height of the two-dimensional R-tree
N	Number of data indexed ($N = N_{3d} + N_{2d}$)
N_{3d}	Number of data indexed in the three-dimensional R-tree
N_{2d}	Number of data indexed in the two-dimensional R-tree
D_{3d}	Density of data indexed in the three-dimensional R-tree
D_{2d}	Density of data indexed in the two-dimensional R-tree

$D_{3d,l}$	Density of node MBBs at level l in the three-dimensional R-tree
$D_{2d,l}$	Density of node MBRs at level l in the two-dimensional R-tree
qk	Average extent of a query rectangle q on dimension k
$s_{,k}$	Average extent of data MBBs on dimension k
t_{start}	represents the minimum of the start times of the data indexed in R_{3d}
$t_{max,3d}$	represents the maximum of the end times of the data indexed in R_{3d}
$t_{min,2d}$	represents the minimum of the start times of the data indexed in R_{2d}: it is the start time of the position that has been held the longest
X_{qt}	$X_{qt} = 1$ if $qt \cap [t_{start}, t_{max,3d}] \neq \emptyset$ and 0 otherwise
q	Query window q = (qx ,qy, qt)
Y_{qt}	$Y_{qt} = 1$ if $qt \cap [t_{min,2d}, CurrentTime] \neq \emptyset$
NA_total(R,q)	Number of node accesses for a selection query between the tree R and the query window q

An extension of the cost model outlined in [TS96] for multi-dimensional range queries has been used to derive an analytical cost formula "*NA_total(2-3TR-tree,,q)*" for our newly proposed 2+3TR-tree. This cost formula is shown below. For additional information the interested reader is referred to [ABGI02].

$$NA\_total(2-3\text{TR - tree}, q) = X_{qt} \sum_{l=1}^{1+\left\lceil \log_f \frac{N_{3d}}{f_{3d}} \right\rceil} \left\{ \frac{N_{3d}}{f_{3d}^{\ l}} \cdot \prod_{k=1}^{3} \left(\left(D_{3d,l} \cdot \frac{f_{3d}^{\ l}}{N_{3d}} \right)^{1/3} + qk \right) \right\} + Y_{qt} \times \sum_{l=1}^{1+\left\lceil \log_f \frac{N_{2d}}{f_{2d}} \right\rceil} \left\{ \frac{N_{2d}}{f_{2d}^{\ l}} \cdot \prod_{k=1}^{2} \left(\left(D_{2d,l} \cdot \frac{f_{2d}^{\ l}}{N_{2d}} \right)^{1/2} + qk \right) \right\} \quad (1)$$

With

$$D_{3d,0} = \sum_{N_{3d}} \prod_{k=1}^{3} sk = N_{3d} \cdot \prod_{k=1}^{3} sk \qquad D_{2d,0} = \sum_{N_{2d}} \prod_{k=1}^{2} sk = N_{2d} \cdot \prod_{k=1}^{2} sk \quad (2)$$

$$D_{3d,l+1} = \left(1 + \frac{D_{3d,l}^{\ 1/3} - 1}{f_{3d}^{\ 1/3}} \right)^3 \qquad D_{2d,l+1} = \left(1 + \frac{D_{2d,l}^{\ 1/2} - 1}{f_{2d}^{\ 1/2}} \right)^2 \quad (3)$$

These expressions specify the query cost for uniform distributions. For non-uniform data distributions, the following modifications have to be performed:

- The average densities $D_{3d,0}$ *and* $D_{2d,0}$ are replaced by the actual densities $D'_{3d,0}$ *and* $D'_{2d,0}$ of the data set within the area of the specified query window *q*.

- The amount N_{3d} and N_{2d} are replaced by a transformation N'_{3d} *and* N'_{2d} computed as follows:

$$N_{3d}' = \frac{D_{3d,0}'}{D_{3d,0}} . N_{3d} \qquad\qquad N_{2d}' = \frac{D_{2d,0}'}{D_{2d,0}} . N_{2d} \qquad (4)$$

$D_{2d,0}$ is the density in the two-dimensional space of the points indexed under the two-dimensional R-tree. $D_{3d,0}$ is the density in the three-dimensional space of the points and segments indexed under the three-dimensional R-tree. The segments correspond to the positions that are hed during several timestamps, so they are all oriented along the time axis.

5. Benchmarking

This section presents the different tests performed to benchmark the 2-3TR-tree. First, we examine its behavior for uniformly distributed datasets and then, for non-uniform datasets. Finally, we compare the 2-3TR-tree to other spatio-temporal indices that have been implemented in [NST99].

We first focus on the parameters in the cost model that are related to the dataset's characteristics, namely N_{3d} and N_{2d}.

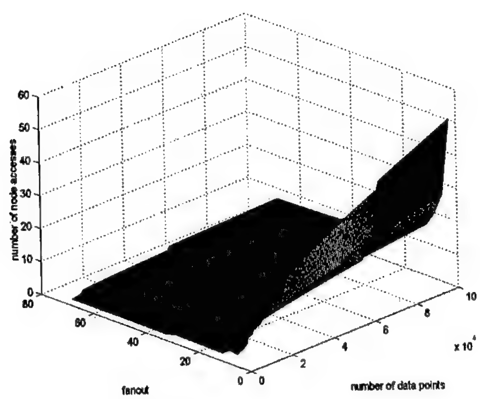

Figure 1: Cost evaluation for N2 = 10% N

In the first series (Figure 1), we also make the fanouts of the two-dimensional and the three-dimensional R-trees vary. Indeed, the fanouts depend on the pages size and the size of the data to be indexed. For practical reasons, we assume that the fanouts of the two-dimensional and

the three-dimensional R-trees are the same: $f_{2d} = f_{3d}$. In this series, the fanouts [2] vary between 3 and 69. The total number of indexed positions, ($N = N_{2d} + N_{3d}$), ranges from 1000 to 100,000, $N_{2d} = 10\%$ N. The query window size is set to 5% of each of the axis, and its time interval overlaps both R-trees. We make the assumption $D_{2d,0} = D_{3d,0} = 0$. This corresponds to densities computed in the context of grids with infinitely small cells (high definition) and involves a lower cost than if we assigned strictly positive densities.

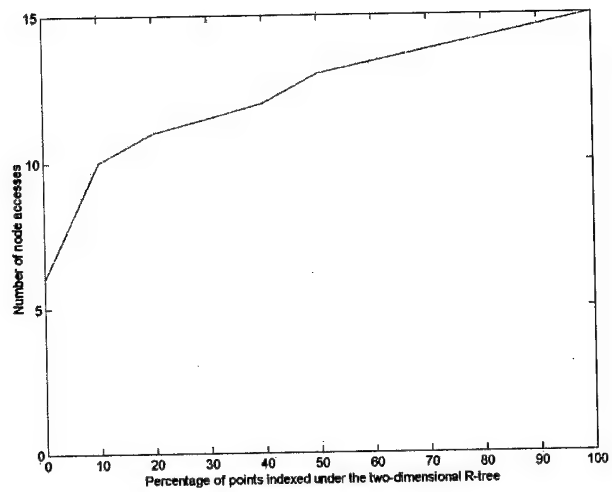

Figure 2: Cost evaluation for N2 varying

The cost drastically increases when the fanout decreases and the number of indexed increases: this is because the height of both R-trees increases.

To obtain more legible and realistic results, f_{2d} and f_{3d} are each assigned a different value (Figure 2). We consider pages of 1024 bytes. Objects characteristics in the three-dimensional R-tree are a set of values (id, x, y, ti, tj), and objects characteristics in the two-dimensional R-tree are a set of values (id, x, y, ti). So, if 4 bytes per number are allocated, for the three-dimensional R-tree M = 1024 bytes/(4 bytes * 5) = 50, and for the two-dimensional R-tree M = 1024 bytes/(4 bytes * 4) = 64. So $f_{2d} = 42$ and $f_{3d} = 33$. The other parameters are set as in the previous experiment.

A larger proportion of points indexed under the two-dimensional R-tree involves more node accesses. This is due to the fact that in the two-dimensional R-tree there is no temporal discrimination. Thus, a range query performed in the two-dimensional R-tree requires the scanning of all the positions that fall within a spatial window and also requires the checking of

[2] 67% M (maximum node capacity) is a typical fanout value for R-trees and variants; here the values taken by the fanouts correspond to M varying from 5 to 103.

whether these positions also fall within the query time interval. Therefore, the processing cost increases.

In this section, we are interested in the influence of the query window size. The query window size is given relatively to the workspace. For instance, $q_x = 0.4$ means that q_x corresponds to 40% of the x-axis. We make the assumptions $D_{2d,0} = D_{3d,0} = 0$, N=100,000 positions and N_{2d} = 10% N, and $f_{2d} = 42$ and $f_{3d} = 33$. We assume that the query timestamps or time interval overlaps both R-trees. This corresponds to one of the worst cases defined in 1.3. The resulting query performance should be lower than if we assumed that the query time interval overlaps only one of the trees. Moreover, for timestamp queries, $q_x = q_y$ = query window size and $q_t = 0$; for time interval queries, $q_t = q_x = q_y$ = query window size.

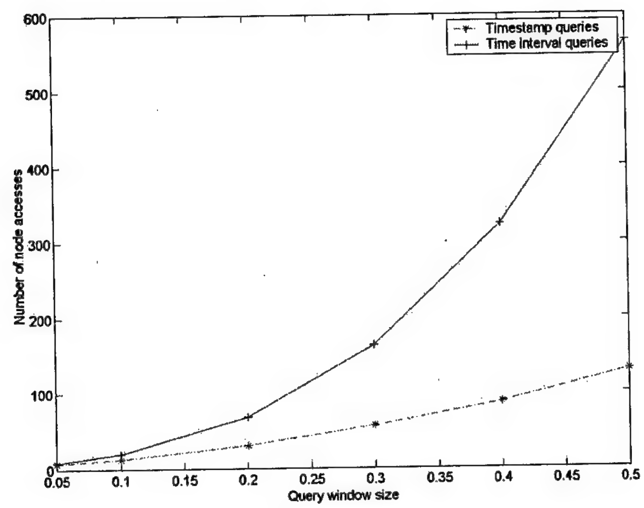

Figure 3: Cost evaluation for q varying and $f_{2d} = 42$ and $f_{3d} = 33$

As expected (Figure 3), the larger the query window, (which corresponds to a cube for time interval queries), the greater number of node accesses. This graph shows high query costs; this is because large query windows (up to 50% of each axis) are considered.

Next, we generate a non-uniformly distributed dataset of points. The workspace is made up of 100*100*100 cells; this defines the granularity. 11,000 positions are indexed: 10,000 under the three-dimensional R-tree and 1,000 under the two-dimensional R-tree. Thus for a uniform distribution, the densities would be defined as follows:

$$D_{2d,0} = 1000/(100*100) = 0.1$$

$$D_{3d,0} = 10000/(100*100*100) = 0.01$$

Several query windows of the same size are computed. For each one we count the number N' of the non-uniformly distributed points that are inside the window, and we derive the density of the points in the query window $D'_{3d,0}$. Then, we evaluate each query cost using: $N'_{3d} = D'_{3d,0} / D_{3d,0} * N_{3d}$ in formula (17). Moreover, we assume that the number of points indexed under the two-dimensional R-tree and to be retrieved by the query corresponds to 10% N'. Finally the costs obtained for all the queries of the same size are averaged and compared to the cost in a uniform distribution.

We now turn our attention to time queries. We first examine point queries, i.e. $q_t = 0$.

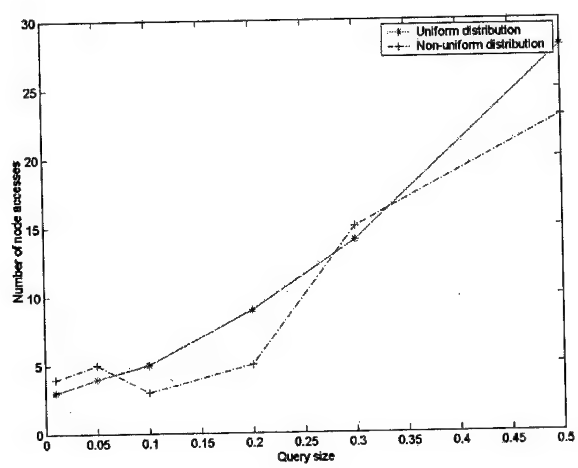

Figure 4: Cost evaluation for timestamp queries in uniform and non-uniform distributions.

As expected (Figure 4), for both distributions the query cost increases with the query window size. Moreover, depending on the densities of the queried regions, the query cost for non-uniform distributions will be lower or higher than the cost for uniform distributions. Indeed, let us consider a non-uniformly distributed dataset with N positions of average density D. According to

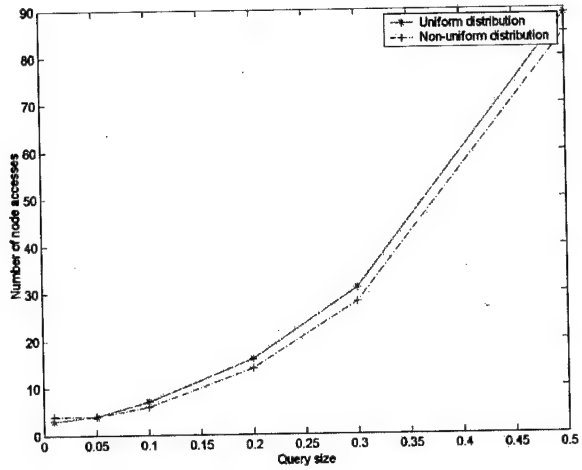

Figure 5: Cost evaluation for time interval queries in uniform and non-uniform distributions.

the model developed in section 4, performing a range query in a region W of density D'>D is equivalent to applying a range query in W to a dataset of N' uniformly distributed positions of density D', where N'=N*D'/D and thus N'>N. Therefore, the resulting query cost is higher than the cost of a range query performed in W and applied to a dataset with N uniformly distributed positions of

density D. The difference between the query costs for uniform and non-uniform distributions increases with the difference in densities within the non-uniformly distributed dataset.

Figure 5 shows the cost evaluation for time interval queries in uniform and non-uniform distributions. The query cost for the non-uniform distribution is slightly lower than that of the uniform distribution. It may be due to the fact that a majority of the query window samples fell within regions of lower densities. We can also notice that the larger the query window, the less relative difference there is between the two costs. Moreover, the standard deviation of the non-uniform distribution query cost samples tends to decrease when the query window size is increased. This can be explained by the fact that the larger the query window size, the closer the average density in this window is to the average density of the dataset (which represents the density of the uniform distribution).

6. Comparison with Other Spatio-Temporal Structures

In this section, the 2-3TR-tree is compared to the 3D R-tree (version3), the 2+3R-tree , and the HR-tree. This comparison is based on the results obtained in [NST99] for uniform distributions. The experiments were run using 100,000 points moving over 100 timestamps. f_{2d} = 42 and f_{3d} = 33. In the cost model for the 2-3TR-tree, we make the assumption $D_{2d,0} = D_{3d,0} = 0$.

Timestamp queries

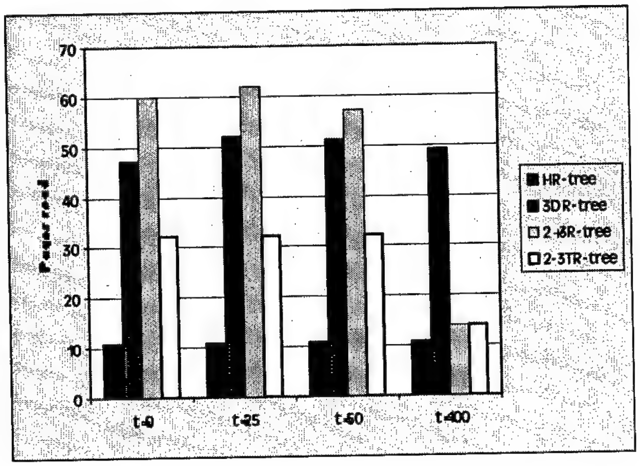

Figure 6: Cost comparison for q = [0.04 0.04 0]

As expected, for all the structures the query cost increases when the query window increases (Figures 6 & 7).

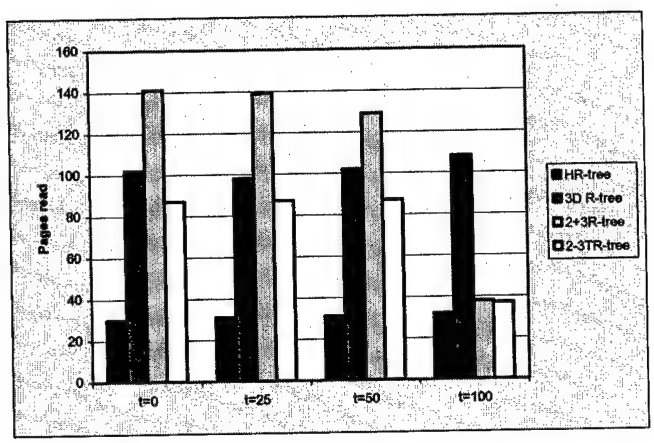

Figure 7: Cost comparison for q = [0.1 0.1 0]

For timestamps queries (Figures 6 & 7), the query costs involved by a structure should be constant for any timestamp, since the distribution is uniform. This is true for all the structures, except for the 2+3R-tree and the 2-3TR-tree: at t = 100, the query cost decreases; this is due to the fact that the three-dimensional of the two R-trees is not traversed because there is no object with an end time equal to 100, thus all the answers come from the 2D R-tree (where there is always a current version for all indexed points).

Even if we consider the 10% error rate of the cost model, the 2-3TR-tree appears to be better than the two other valid-time structures, the 3D R-tree and the 2+3R-tree. This good performance of the 2-3TR-tree as compared to the 2+3R-tree can be explained by its reduction of overlapping.

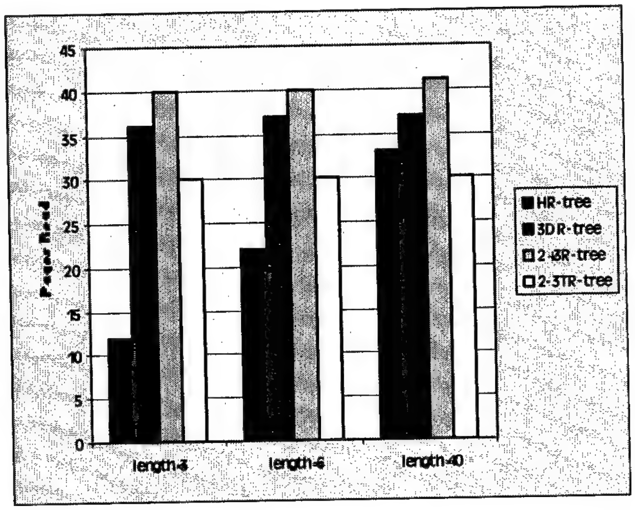

Figure 8: Cost comparison for q = [0.04 0.04 length/100]

However, the HR-tree requires substantially smaller query processing time in all cases. Indeed, when the temporal part of the query is a point, the tree that corresponds to that timestamp is obtained and the query processing is exactly the same one of a containment query in a single standard R-tree. On the other hand, for the 3D, 2+3 R-tree, and the 2-3TR-tree, the whole structure is involved in the query processing, thus increasing significantly the query processing cost.

<u>Time interval queries</u>

Length represents the length of the time interval of the query and is measured in number of consecutive timestamps.

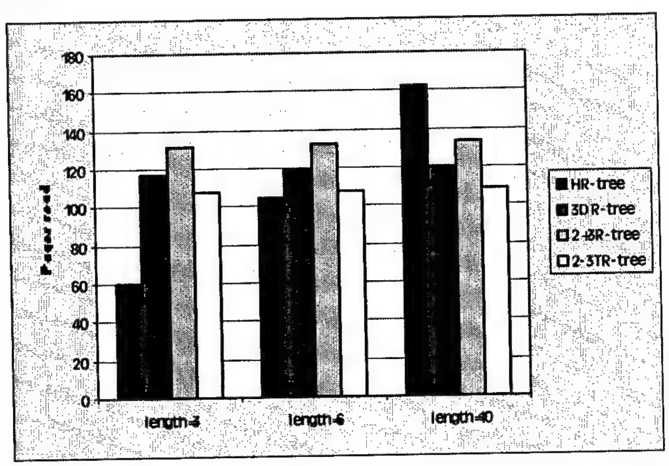

Figure 9: Cost comparison for q = [0.1 0.1 length/100]

Once again, as anticipated, for all the structures, the query cost increases with the spatial range and the time interval of the query (Figures 8 & 9).

We can notice that the HR-tree loses its relative advantage relatively fast with the increase in the queried interval length. Indeed, the HR-tree is well suited to search a time point, for time intervals it has to traverse as many logical R-trees as many indexed time points are covered by that interval. This phenomenon is worsened by the increase in the query window area.

Moreover, the 2+3R-tree loses its advantage as compared to the 3D R-tree because its two-dimensional R-tree is searched and this latter does not provide temporal discrimination.

Query costs involved by the 2-3TR-tree remain lower than those of the 3D R-tree. This can be due to the error involved by the cost model and to the fact that the 2-3TR-tree reduces overlapping as compared to the 2+3R-tree and the 3D R-tree version 3.

From these tests, the HR-tree has better timestamp query performance than the 2-3TR-tree; however, it loses its advantage when the query time interval increases, and it cannot handle valid-time databases. It also appears that the 2-3TR-tree performs range queries at least as well as the two other valid-time index structures, namely the 2+3R-tree and the 3D R-tree version 3. Moreover, the 2-3TR-tree supports dynamic data acquisition, unlike the 3D R-tree and the HR-tree, and handles trajectory retrievals, whereas none of the other tested structures does.

7.Conclusion

This paper first presented existing access methods for spatial data. From this discussion, we pointed out that the R-tree and its derivatives were the most efficient structures for indexing non-uniformly distributed data. That is why we then focused on and described spatio-temporal indices based on R-trees.

A new taxonomy that classifies spatio-temporal access methods regarding to dataset characteristics and query requirements was defined. This new taxonomy classifies any existing R-tree structure as a point in a 8-dimensional space. According to this taxonomy, we specified the requirements involved by a type of applications that could not be handled by any existing spatio-temporal index structure. We then designed a spatio-temporal access method, the 2-3TR-tree, based on the 2+3R-tree and the TB-tree that meets these requirements. More particularly the 2-3TR-tree handles valid-time and fully evolving datasets; it supports dynamic data acquisition; it uses a minimum overlap or a linear ordering insertion/split strategy and offers spatio-temporal discrimination: therefore, it efficiently retrieves historical information; moreover, it does not involve version duplication.

Finally, in order to benchmark this new structure, an analytical cost model based on an existing cost model for R-trees was defined. The 2-3TR-tree was compared to other spatio-temporal indices and its behavior was analyzed. From these tests, the 2-3TR-tree appears to perform well in historical information retrievals as compared to the HR-tree,

the 3D R-tree version 3, and the 2+3R-tree. Moreover, it handles trajectory retrievals whereas the other tested structures do not.

8. Acknowledgments

The work of Kevin Shaw and Roy Ladner was supported in part by Naval Research Laboratory's Base Program, Program Element No. 0602435N. Special thanks to Yannis Theodoris, Mario A. Nascimento, and Jefferson R.O. Silva who provided us with valuable information regarding their cost model for R-tree structures as well as their experiments on spatio-temporal indices.

9.References

[ABGI02] Abdelguerfi, M., Givaudan, J, "Advances in Spatio-Temporal R-tree Based Structures", Technical Report TR012-02, Computer Science Department, University of New Orleans, 2002.

[BECK90] Beckmann, N., H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree; An efficient and Robust Access Method for Points and Rectangles", In *Proc.ACM SIGMOD International Conference on Management of Data*, pp.322-331, 1990.

[FIBE74] Finkel, R.A. and J.L. Bentley, "Quad trees: A Data Structure for Retrieval on Composite Key", *Acta Inform*, 4:11-9, 1974.

[GUTT84] Guttman, a., "R-trees: A Dynamic Index Structure for Spatial Searching", In *Proc.ACM SIGMOD International Conference on Management of Data*, pp.47-57, 1984

[KAFA94] Kamel, C. Faloutsos, "Hilbert R-tree: An Improved R-tree Using Fractals", Proceedings of the 20 th Conference on Very Large Data Bases (VLDB), pp. 500-509, 1994.

[NAS98] M.A. Nascimento and J.R.O. Silva, "Towards historical R-trees", In Proc. of the 1998 ACM Symposium on Applied Computing, pages 235 - 240, February 1998.

[NST99] Nascimento, M.A., Silva, J.R.O and Theodoridis, Y., "Evaluation of Access Structures for Discretely Moving Points". In Proc. of the Intl. Workshop on Spatiotemporal Database Management (STDBM'99), pp. 171-188. Edinburgh, UK, Sep/99.

[PJT00] Pfoser, D., Jensen, C., Theodoridis, Y. "Novel Approaches to the Indexing of Moving Object Trajectories". *VLDB*, 2000.

[LASA02] Roy Ladner, Kevin Shaw, Mahdi, Abdelguerfi (editors), Mining Spatio-Temporal Information Systems, in print, Kluwer Academic Press, 2002.

[SELL87] Sellis, T., N. Roussopoulos, and C. Faloutsos, "The R+-tree : A dynamic index for multidimensional objects", In *Proc. 13th Int. Conf. on Very Large Databases*, pp.507-518, 1987.

[TS96] Y. Theodoridis, T. Sellis, "A Model for the Prediction of R-tree Performance", In *Proceedings of the 15th ACM Symposium on Principles of Database Systems* (PODS), 1996.

[TS98] Yannis Theodoridis, Timos Sellis, Apostolos N. Papadopoulos, Yannis Manolopoulos, "Specifications for Efficient Indexing in Spatiotemporal Databases", *In Statistical and Scientific Database Management*, 1998.

[TZO98] T. Tzouramanis, M. Vassilakopoulos, and Y. Manolopoulos, "Overlapping linear quadtrees: A Spatio-Temporal Access Method", In Proc. of the 6th ACM Intl. Work-shop on Geographical Information Systems, pages 1- 7, November 1998.

[VAZ96] Y. Theodoridis, M. Vazirgiannis, and T. Sellis, "Spatio-Temporal Indexing for Large Multimedia Applications. In Proc. of the 3rd IEEE Conf. on Multimedia Computing and Systems, pages 441 - 448, June 1996.

[XULU90] X. Xu, J. Han, and W. Lu, "RT-tree: An improved R-tree index structure for spatiotemporal databases", In Proc. of the 4th Intl. Symposium on Spatial Data Handling, pages 1040 - 1049, 1990.

[YUPA00] T. Yufei, D. Papadias, " MV3R-tree: A Spatio-Temporal Access Method for Timestamp and Interval Queries", *Technical Report HKUST-CS00-06*, 2000.

USEFUL HURRICANE INFORMATION

HURRICANE ANNOUNCEMENTS:

1. When authorized, CNMOC and NASA, will issue a joint announcement instructing employees when Stennis Space Center will be closed or reopened for normal working hours. This information will be broadcast on most radio and television stations within the SSC commuting area. Specific information on the status of NRL-SSC will be available by dialing 1-228-688-4010. Information is also available by dialing into the NASA Emergency Operations Center, 1-228-688-3777.
2. Personnel should also listen to the following TV and radio stations for reporting to work instructions:

RADIO (AM Frequencies)	RADIO (FM Frequencies)	TELEVISION
WVMI 570 - Biloxi	WQID 93.1 - Biloxi	WWL 4 - New Orleans
WWL 870 - New Orleans	WKKN 97.9 - Biloxi	WLOX 13 - Biloxi/Gulfport
WBSL 1190 - Bay St. Louis	WGCM 101.9 - Gulfport	WXXV 25 - Gulfport
WRJW 1320 - Picayune	WJOJ 106.3 - Picayune	WDSU 6 - New Orleans
WSDL 1560 - Slidell	WXHR 106.9 - Slidell	WKRG 5 - Mobile

Recent experience indicates that WWL 870 AM is the most listened to radio station for weather information throughout the local area. However, SSC information is broadcast more frequently by the local community stations.

HURRICANE PREPARATIONS:

1. When a hurricane threatens, all employees should:

    - Keep abreast of official weather reports.
    - Read your newspaper and listen to radio and television for official announcements.
    - Note the location of the nearest Emergency Shelter.
    - Pregnant women, the ill and infirm should call a physician for advice.
    - Be prepared to turn off gas, water and electricity where they enter your home.
    - Fill tubs and containers with water, typically one gallon - person - day.
    - Make sure your car has a full tank of gas.
    - Secure your boat. Use long lines to allow for rising water.
    - Secure moveable objects on your property.
    - Board up or tape windows and glassed areas. Remove furniture, and other lawn products, in the vicinity of windows.

HURRICANE EVACUATION:

- Leave as soon as you can, follow official instructions only.
- Disconnect all electrical appliances except refrigerator and freezer. Their controls should be turned to the coldest setting and the doors kept closed.
- Leave food and water for pets. Seeing eye dogs are the only animals allowed inside the shelters.

STOCK ADEQUATE SUPPLIES OF:

Batteries (and radios)	Hand Tools (Hammers, Pliers)
Flashlights	Tape

First-aid kit
Candles and matches
Prescription medications
Disinfectant
Insect Repellent
Plastic drop cloths
Boards
Rope **
Ax **
Containers for water
Water purification tablets
Can opener and utensils
Canned meat
Canned food, juices, soft drinks
Gum, candy
Charcoal bucket and charcoal
Fire extinguisher
Buckets of sand
Life jackets
Hard-top head gear
Ready cash

** Keep ax to cut an emergency escape opening. If you go to the upper floors or attic of your home, take rope for escape to lower levels when water subsides.

•

USEFUL WEB SITES:

Louisiana State Emergency EVACUATION ROUTE MAPS:

http://www.dotd.state.la.us/maps/state.pdf

Mississippi Emergency Management Agency information web page:

http://www.memaorg.com/hurricanepreparedness.htm

Southern Mississippi county EVACUATION ROUTE MAPS:

http://www.memaorg.com/hancockcountyevac.jpg
http://www.memaorg.com/harrisoncountyevac.jpg
http://www.memaorg.com/jacksoncountyevac.jpg